

Rapport du Projet de conception

Téléguider un robot grâce à la WIIFIT

Merlaud Pierre
Foughali Omar
Grira Abdelkarim
EL hajji Khalil

Juin 2010

Sommaire :

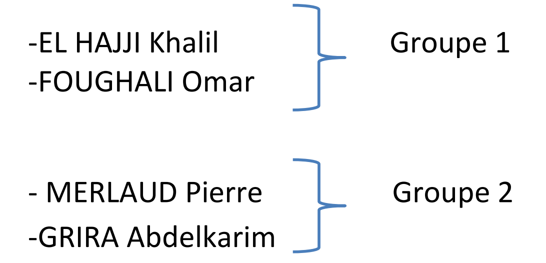
a. Introduction :

Nos motivations

En premier temps nous aborderons l'aspect matériel de notre projet, dans lequel nous allons détailler les différents composants utilisés, les problèmes relatifs aux matériels ainsi que le montage du robot, en deuxième temps nous parlerons de la partie logicielle, divisée à son tour en deux sous parties. Dans la première nous expliquerons avec précision les différentes portions du code utilisées sous « Visual Basic express 2008 », concernant la deuxième sous partie, nous l'avons réservée à l'explication du programme en « C » qui commandera le robot grâce à la carte Arduino. Pour Finir , nous énoncerons tous ce que nous a apporté ce projet autant sur le plan organisationnelle que sur le plan technique , nous proposerons aussi quelques idées de projets pour l'année prochaine , sans oublier les différents problèmes rencontrés tout au long de ce projet.

Le choix de ce projet est loin d'être aléatoire, notre volonté à relever les défis, notre intérêt à la découverte ainsi que l'originalité du projet nous ont poussé à choisir ce dernier. Mener à bien ce projet nous aidera à concrétiser nos connaissances théoriques acquises durant le cycle préparatoire, d'autant plus que ce projet s'inscrit dans le domaine de l'automatique et du génie informatique qui nous intéresse particulièrement.

Pour mener à bien ce projet sans perte de temps et avec efficacité, nous nous sommes partagés en deux groupes :



FOUGHALI Omar et EL HAJJI Khalil avaient une préférence pour le langage VB , alors que MERLAUD Pierre et GRIRA Abdelkarim avaient une préférence le langage C. Pour le bon déroulement de notre projet, chacun a choisi le langage où il se sentait alaise, pour ainsi être le plus productif possible.

Le groupe 1 s'est chargé de la parie logiciel en VB faite sur PC.

Le groupe 2 s'est chargé de la partie logiciel en C sur la carte « Arduino Mega ».

On détaillera avec précision le travail de chaque groupe et son utilité dans le chapitre «Logiciel »

Présentation du projet

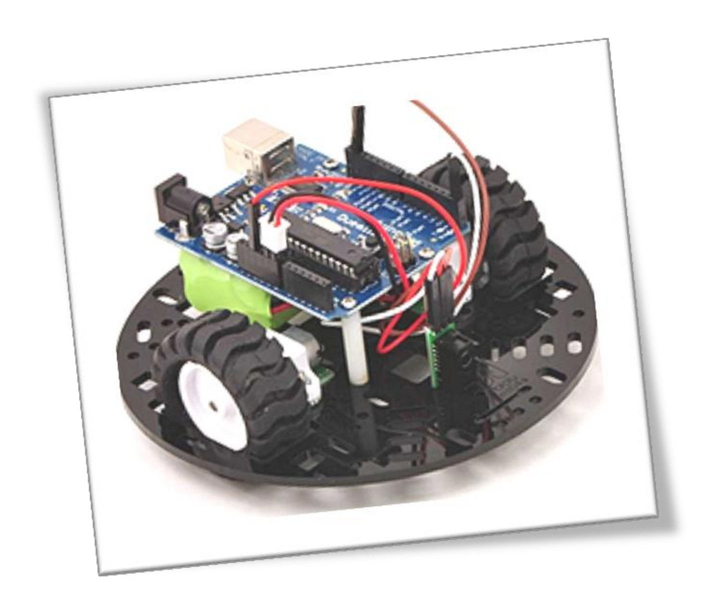


L'originalité du projet consiste dans le fait qu'on utilise notre corps pour commander un robot et tout cela sans fil bien sûr .Expliquons un peu tout ce qui se cache derrière cette magie :

Quand une personne se me debout sur la WII fit , les quatre capteurs de pression présents sur cette dernière détecte différentes pressions selon la position du corps par rapport au centre de de la WII fit. Par exemple si le corps est penché devant les capteurs avant détectent une plus grande pression que ceux de derrières. Les valeurs données par chaque capteurs sont envoyé en continue à l'ordinateur via un signal Bluetooth. Grace à un récepteur Bluetooth branché sur le PC , nous récupérons les valeurs relevées par les quatre capteurs , puis nous fessons des calculs qu'on détaillera plus tard , pour n'envoyer ensuite que deux valeurs pour commander les deux moteurs de notre robot. La communication entre le robot et le PC se fait grâce à deux carte FM « Radiometrix » , une connecté sur le PC et la deuxième branché sur le robot. C'est donc ainsi qu'opère notre magie.

b. Matériel

Châssis du robot

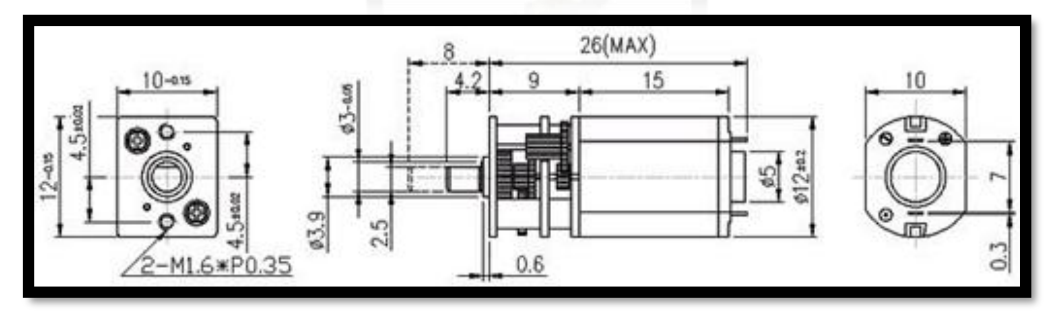


La base du robot est constituée :

- D’un **châssis** *Pololu*, qui consiste en un disque de matière plastique usiné par laser avec un grand nombre de perçages, permettant de fixer un module Arduino par exemple (sur la photo ci-dessus c’est une carte *Duemilanove*, mais dans notre cas c’est une *Mega*).

- De **2 moteurs/réducteurs** à courant continu de petite taille (24 x 10 x 12 mm), dotés d’un rapport de réduction de 30 :1, qui peuvent être alimentés de 3 à 7 V et fournir une vitesse de 440 tpm (pour une alimentation de 6 V).





- D’**une paire de roues** (Dimensions: Ø 42 x 19 mm).



- De **2 roues folles**, une à l’avant et une à l’arrière, pour assurer l’équilibre du robot (hauteur de 10 cm, entraxe de fixation : 13,2 mm).



Carte Arduino Mega



La « *carte mère* » du robot Wii’Stia est la carte Arduino *Mega*.

C’est un module Arduino construit autour d'un microcontrôleur(dans notre cas un ATmega1280), et de composants complémentaires qui facilitent la programmation et l'interfaçage avec d'autres circuits.

Le microcontrôleur est pré-programmé avec un *bootloader* de façon à ce qu'un programmateur dédié ne soit pas nécessaire.

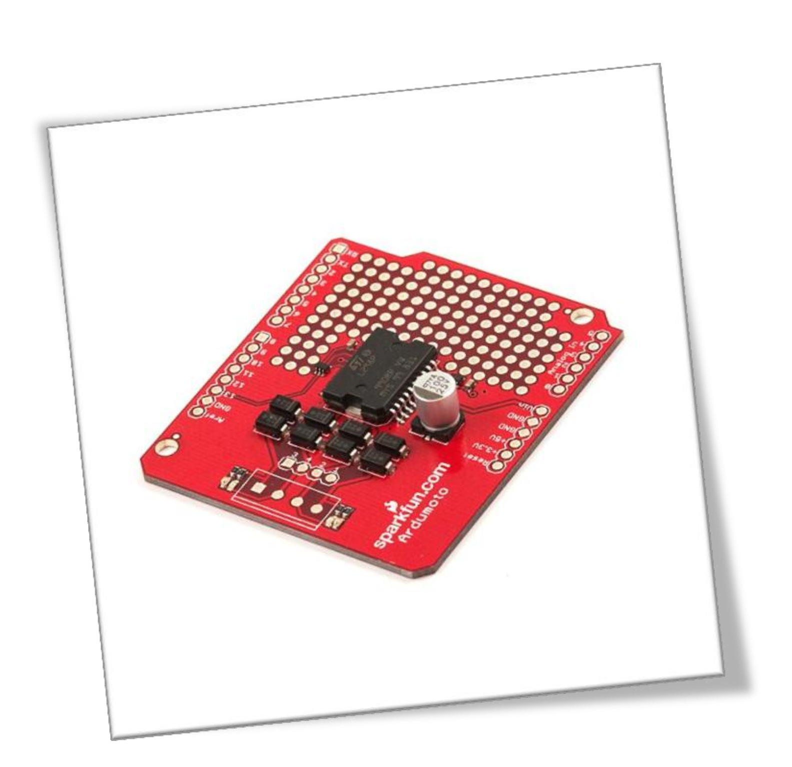
La carte se programme au travers d'une connexion série RS-232, mais les connexions permettant cette programmation diffèrent selon les modèles. Dans le cas de la *Mega*, c'est un port USB.

L'Arduino utilise la plupart des entrées/sorties du microcontrôleur pour l'interfacage avec les autres circuits.

Notre modèle possède 54 entrées/sorties numériques (dont on se sert grâce aux fonctions pinMode(), digitalWrite(), anddigitalRead()), dont 14 peuvent produire des signaux PWM (permettant d'obtenir une tension continue variable à partir d'une source de tension continue fixe), et 16 entrées analogiques.

Les connections se font par les connecteurs femelles situés sur le dessus de la carte. Les modules d'extension viennent s'empiler sur l'Arduino (par exemple la carte *Ardumoto* sur Wii'Stia).

Carte Ardumoto



Cette carte, qui vient s'empiler sur la carte principale *Mega*, est ce qui nous permet de contrôler les 2 moteurs à courant continu de notre robot.

Elle s’alimente à partir de la même ligne VIN que la carte Arduino. Elle dispose de LEDs bleu et jaune, qui indiquent la direction active (sens de rotation) des moteurs.

La commande moteur fixée à OUT1/2 est connectée à la ligne numérique 12 (direction A) et à la ligne numérique 10 (MID A).

La commande moteur fixée à OUT3/4 est connectée à la ligne numérique 13 (direction B) et à la ligne numérique 11 (MID B).

Carte Radiometrix (ondes radio)

- Un module "TDL2A-433" (9600 bds)
- Un circuit d’interfaçage MAX232
- Un connecteur SUB-D 9 broches
- Une led de visualisation (permettant de valider la réception de données)
- Un étage de régulation
- Un connecteur pour alimentation externe via une pile 9 V
- Une antenne avec raccordement sur connecteur SMA.



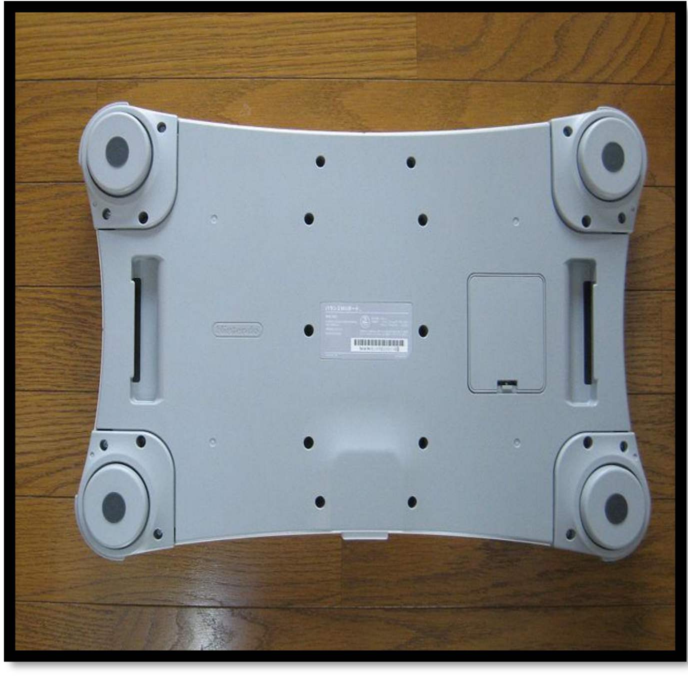
2 platines "TDi2-433" pourront être utilisées au moyen d’un terminal de saisie (type HyperTerminal™ par exemple) pour effectuer des essais de transmission de fichiers entre PC. Il vous sera également possible d'effectuer des tests de communication entre 2 platines "TDi2-433" ou entre une platine "TDi2-433" et un ou plusieurs modules "TDL2A-433".

Cette carte sera utilisée pour envoyer les informations au robot. On va donc en utiliser 2 : une sera installée sur le robot et l’autre liée au pc (logiciel de contrôle).

Wii Board

La Wii Balance Board est un accessoire en forme de pèse-personne électronique pour la console de jeux vidéo Nintendo Wii. Elle a été dévoilée pour la première fois le 11 juillet 2007 à l'E3 en combinaison avec le logiciel Wii Fit.





Principe de fonctionnement :

Elle nécessite l'utilisation de 4 piles de type AA pour fonctionner. Son autonomie annoncée est de 60 heures.

La balance n'utilise pas de fils pour se connecter à la console. Elle est constituée de plusieurs capteurs de pression utilisés pour mesurer le centre des pressions des pieds de l'utilisateur, donc grâce à 4 capteurs de poids dans la Balance, on peut calculer l'inclinaison du joueur et son centre de gravité. Dans une entrevue avec le site IGN Entertainment, Il s'est avéré que les capacités de mesure de la Wii Balance Board étaient probablement supérieures à celles des pèse-personnes électroniques classiques.

La Balance est autorisée pour des personnes pesant jusqu'à 150 kg.
Dans le manuel d'utilisation, il est dit que la Wii Balance doit être utilisée sur des surfaces stables et que son utilisation sur une moquette ou un tapis épais peut créer disfonctionnement.

Dimensions : 511 X 316 X 53.2

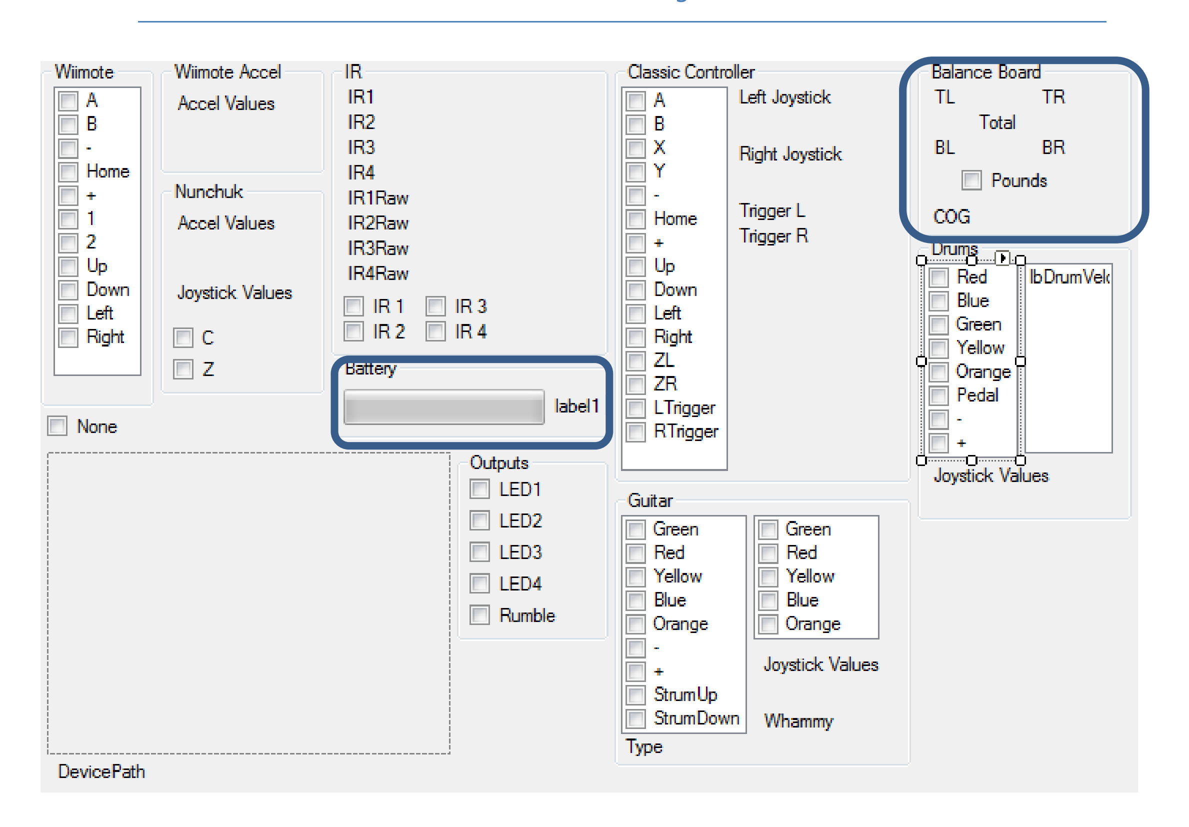
c. Logiciel

i. Programmation en VB express 2008

1. Prise en main et premiers essais.

Avant de se lancer dans la rédaction du code pour l'application définitive , nous avons d'abord décortiquer notre projet en plusieurs missions. Ensuite nous avons essayé d'accomplir chaque mission séparément.

-Récupération de l'application de « Codingforfun.com » , qui gère la communication entre la WIIFit et le PC→ alléger le code



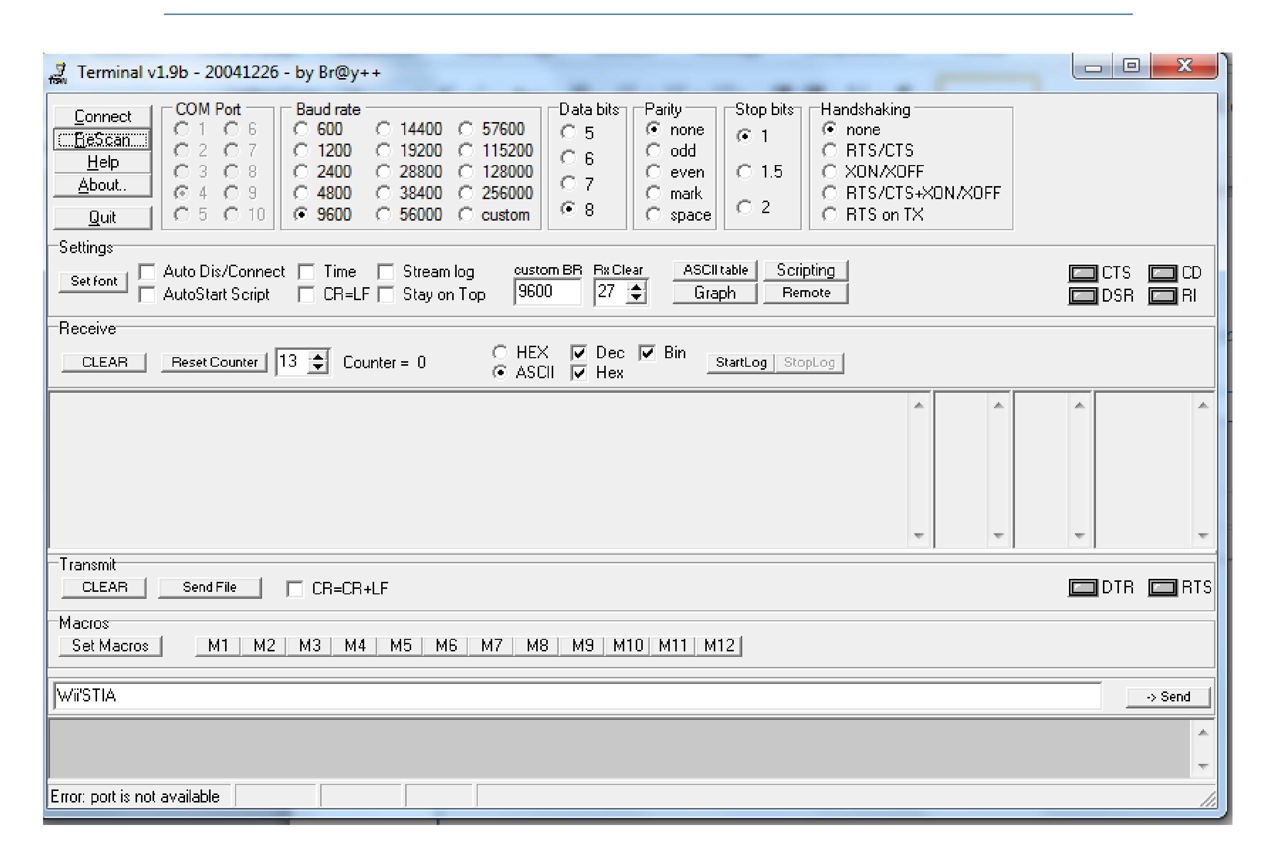
capture d'écran du programme avant nos modifications

Le programme que nous avons récupéré depuis le site www.codingforfun.com pouvait gérer la communication entre tous les accessoires de la WII et le PC, or que nous utilisons qu'un seul, la WIIFit.

Pour alléger le programme, il ne fallait pas juste supprimer les parties du code concernant les autres accessoires , mais aussi fait attention au variables utilisées dans différentes parties du code, pour ne conserver que les parties vitales du code.

Il fallait bien sûr comprendre code, et assimiler sa logique afin de réutiliser convenablement les parties qui nous intéressent , qui sont encadrées sur la capture d'écran ci-dessus .

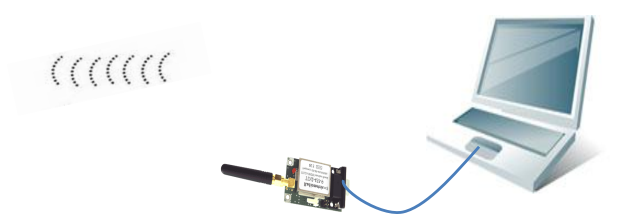
-Découvrir la communication entre 2 PC grâce aux ports RS32→ tester les cartes sans fils.

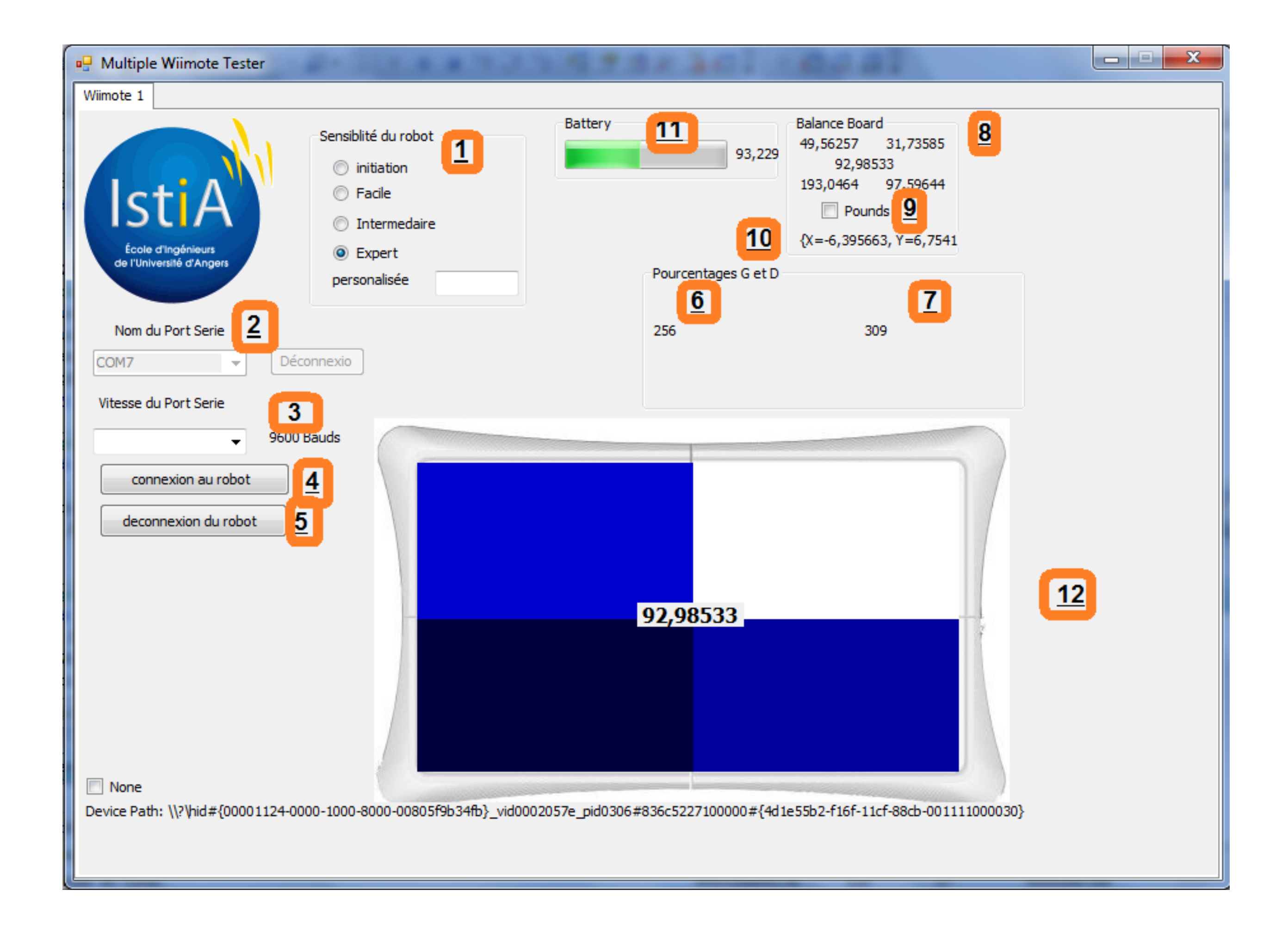


Avant de créer notre propre programme qui communiquera avec le robot « WII'STIA », nous avons découvert la communication entre deux PC grâce aux ports RS32 en utilisant le logiciel «Terminal». Ce programme nous permet d'envoyer et de recevoir des chaines de caractères d'un PC à autre en utilisant les cartes sans fils « radiometrix ».On a découvert plusieurs caractéristiques d''un signal passant par un port RS32 , à savoir :

- Le débit
- La parité
- Bits d'arrêts
- Nombre de bits transmis
- Handshaking







2. *Explication du code*

1) **Sensibilité du robot** : ici on peut changer la vitesse maximale des deux pourcentages qu'on calcule à partir des pressions sur la balance board, du coup on change la sensibilité des deux moteur qui dirigent les deux roues.

2) **Choix du port série** : Il s'agit d'une ComboBox qui détecte tout les ports séries disponibles dans le pc et donne à l'utilisateur le choix entre ces derniers et s'y connecter à l'aide d'un bouton qui sert aussi de bouton déconnexion.

3) **Choix de la vitesse du port** serie : Il s'agit d'une combobox qui mets à la disposition de l'utilisateur le choix entre plusieurs vitesse du port série (la valeur de 9600 est mise automatiquement si aucune vitesse n'est choisie)

4) **Bouton connexion au robot** : Ce bouton déclenche le Timer qui envoie les pourcentages de pressions toutes les 100 ms.

5) **Bouton déconnexion du robot** : Ce bouton arrête le Timer le robot ne reçoit donc plus d'informations puis on envoi (355 & 355) pour l'arrêter

6) **Pourcentage gauche** : Il s'agit du pourcentage qu'on calcule à partir des deux pressions qui correspondent à la partie gauche de la balance board, et qui correspond aussi à la vitesse et la direction de la roue gauche.

7) **Pourcentage Droite**: Il s'agit du pourcentage qu'on calcule à partir des deux pressions qui correspondent à la partie droite de la balance board, et qui correspond aussi à la vitesse et la direction de la roue droite.

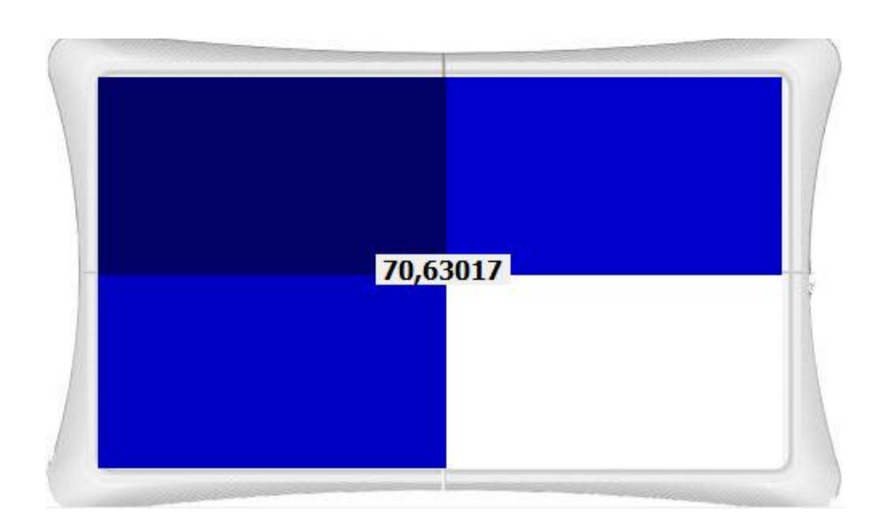
8)**Information de la balance board** : Ici on expose les quatre pressions sur les capteurs de la balance board et le poids de l'utilisateur si il est dessus.

9) **Unité** : il s'agit d'une checkboxe qui permet de passer du kilogramme au pound.

10) **Centre de gravité de la balance board** : ce label donne les coordonées du centre de gravité de la balance board.
11) **Batterie de la balance board** : Il s'agit d'un affichange qui montre l'état de la batterie de la balance board.

12)**Simulation du fonctionnement de la balence board** : Il s'agit d'une simulation de la balance board avec la clarté de la couleur qui change en fonction de la pression sur les capteurs de la balence board.

Analyser les valeurs des différents capteurs de la Wii fit :

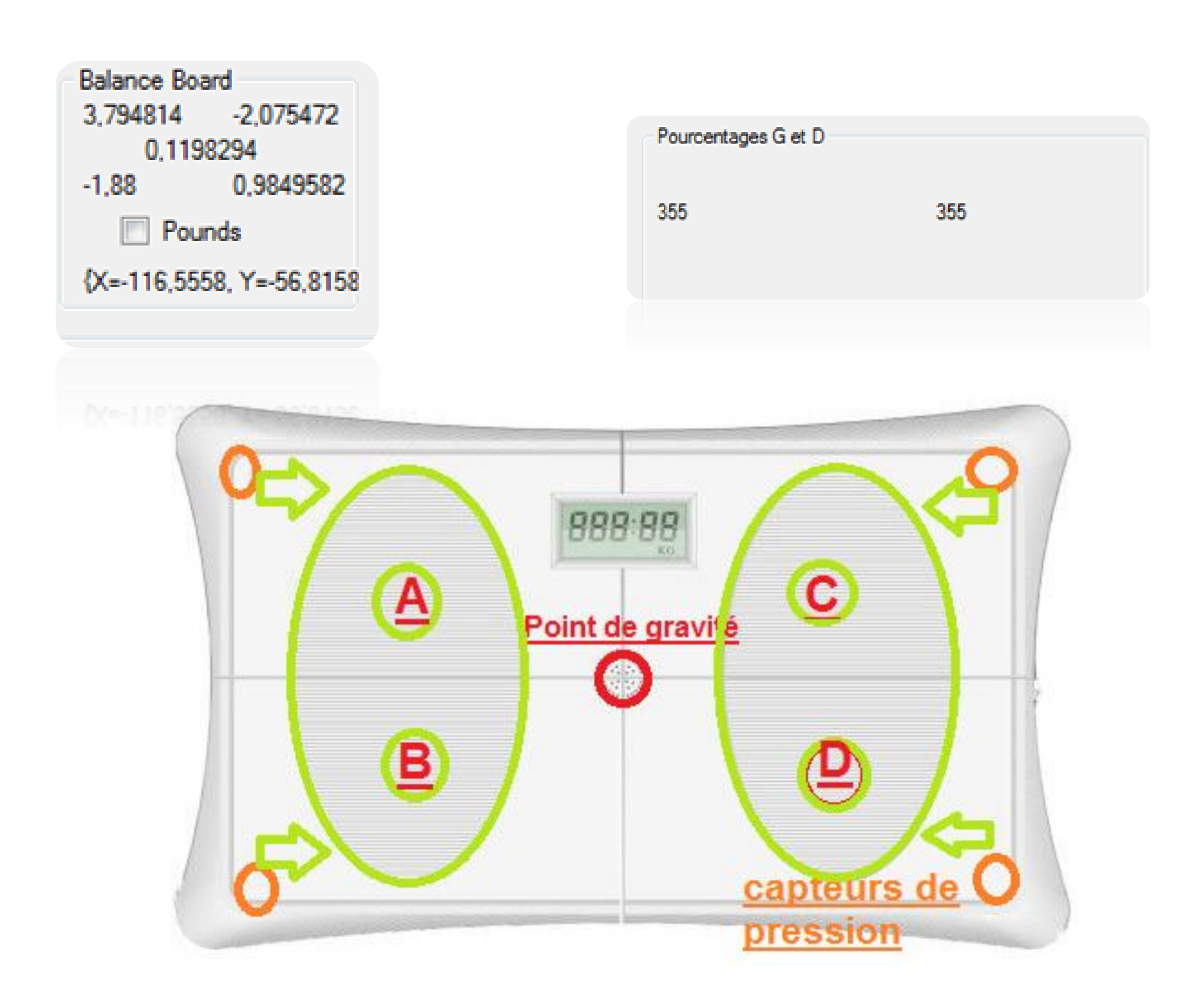


- *Affichage de wiifit sur l'écran avec un changement de l'intensité en fonction de la pression exercée sur chaque capteur*

On a décidé de représenter l'intensité de la couleur bleu de en fonction de la pression appliqué sur chaque capteur , pour aider à mieux trouver notre équilibre et diriger notre robot .

- *Explication des différents calculs*

Pour commencer , on a d'babord ignorer les valeurs dues à l'impressision des capteurs de la WIIFIT quand personne n'est dessus.C'est-à-dire annuler les valeurs qui sont proche du zéro , pour ne pas faire bouger le robot quand personne n'est sur la WIIFIT.(on expliquera plus blus bas pourquoi on envoi 355 et non pas 0).



Comme Indique la photo ci-dessus chaque capteur de pression nous donne une valeur (A,B,C,D) à partir desquels ils faudra calculer 2 pourcentages qui vont définir la vitesse et le sens des 2 roues du robot .

Les 2 pourcentages :

Pourcentage pour la roue droite : (A-B)/(A+B+C+D)

Pourcentage pour la roue gauche : (C-D)/(A+B+C+D)

Ce qui nous donne des valeurs comprises entre -1 et 1. Les valeurs négatives feront tourner la moteur dans le sens inverse de celui des valeurs positives.

Ensuite on multiplie par 255 puis on décale les valeurs de 355 vers la droite.

On a choisi de multiplier par 255 pour faciliter l'interprétation des valeurs reçu par le robot, puisque les moteurs tournent sur une plage de [-255,255].Puis on ajoute 355 aux valeurs calculées pour n'envoyer que des nombres à trois chiffres compris entre 100 et 610.Dans ce cas notre **zéro** devient le **355.**

Dans notre logiciel on propose de donner le choix à l'utilisateur de choisir les ports serie disponibles sur son pc.

Pour s'y faire on utilise la fonction « `IO.Ports.SerialPort.GetPortNames()` » de vbexpress qui détecte tout les ports series disponibles et après on les mets dans un tableau de chaine de chasractères.

Configuration du port RS232 :

- pas de parité
- un bit d'arrêt par octet
- nombre de bit par octet
- ligne Rts activé : pour envoyer des données
- ligne Dtr activé : pour recevoir des données

Enfin, on mets tout les ports séries trouver dans une combobox. Et si on trouve pas on affiche un message et on ferme l'application.

Connexion au port série :
Après le choix du port et la vitesse, l'utilisateur doit se connecter ,pour cela il faut :

- ouvrir le port désiré à l'aide de la fonction « RS232.Open() » (envoie un message d'erreur si le port est déjà utilisé)

- Si le port fonctionne on se connecte au port en affichant la vitesse du port et en désactivant le combobox pendant l'utilisation du port.

- Si le port ne marche pas on efface la vitesse, on ferme le port et on réactive le combobox.

Envoyer les données au robot :

Après la connexion au robot on passe à la connexion au robot c'est-à-dire l'envoie des données à la carte du robot, pour cela il faut :

- Convertissant les 2 pourcentages des moteur en char, puis on les envoie en utilisant la fonction « RS232.Write(pourg & pourd)»
- Mettant le code dans un Timer d'intervalle : 100 ms.

Donc l'utilisateur n'a qu'à se connecter à l'aide du bouton connexion qui active le timer.

ii. Programmation en en C (Arduino)

1. Prise en main et premiers essais.

Programmation du robot : Arduino



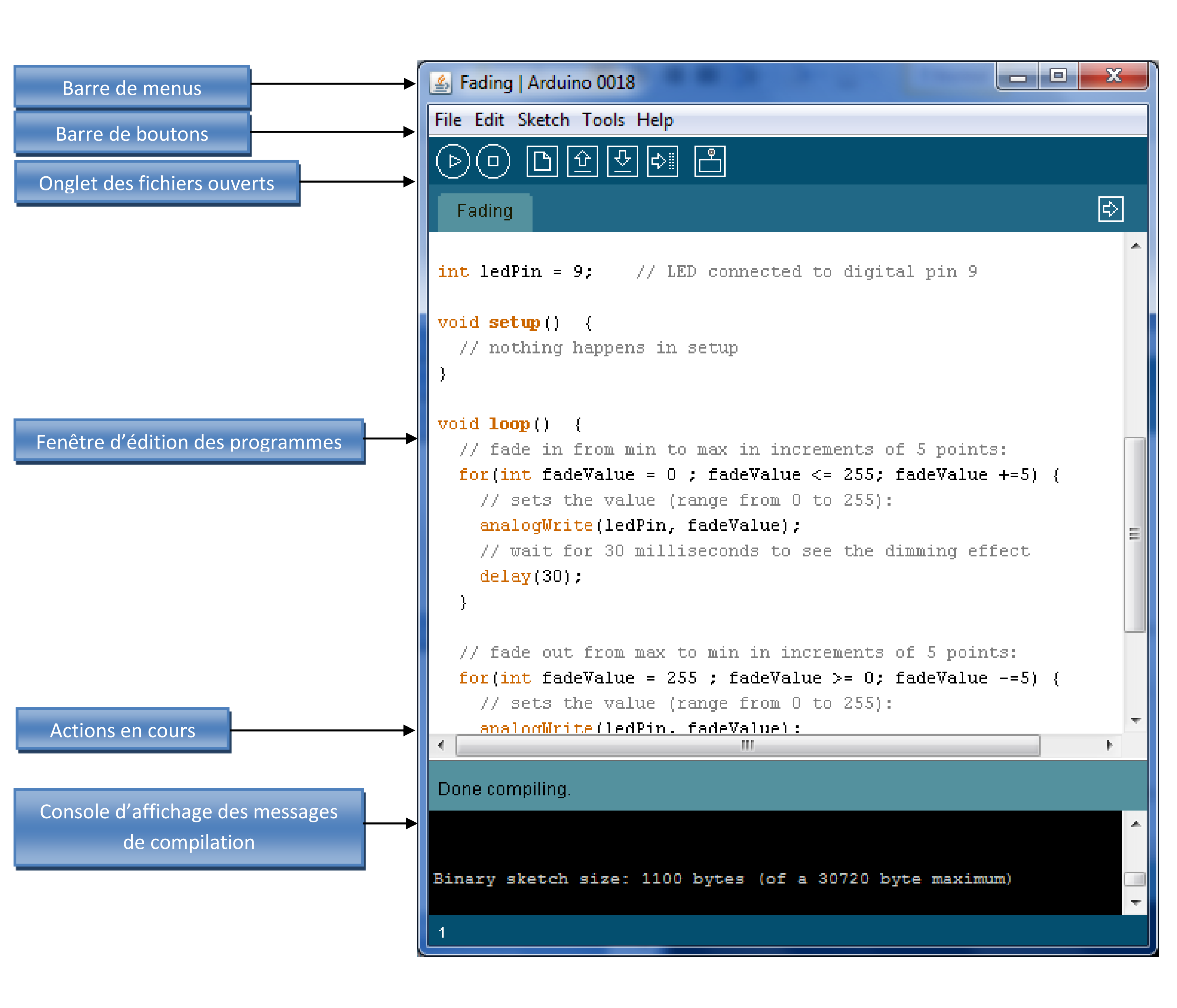
Nous avons utilisé le logiciel ***Arduino*** (disponible sur le site officiel d'Arduino : http://arduino.cc) pour programmer Wii'Stia. Cette application Java open-source fonctionne sur tous les systèmes d'exploitation et permet de programmer des modules Arduino (dans notre cas la carte Arduino Mega). Ce logiciel permet :

- D'écrire et de compiler des programmes pour la carte Arduino,
- De se connecter avec la carte arduino pour y transférer nos programmes,
- Et de communique avec la carte.

Le langage de programmation utilisé est une variante *« allégée »* du C et du C++, restreinte à l'utilisation d'un module Arduino, de ses entrées, sorties, et des librairies associées.

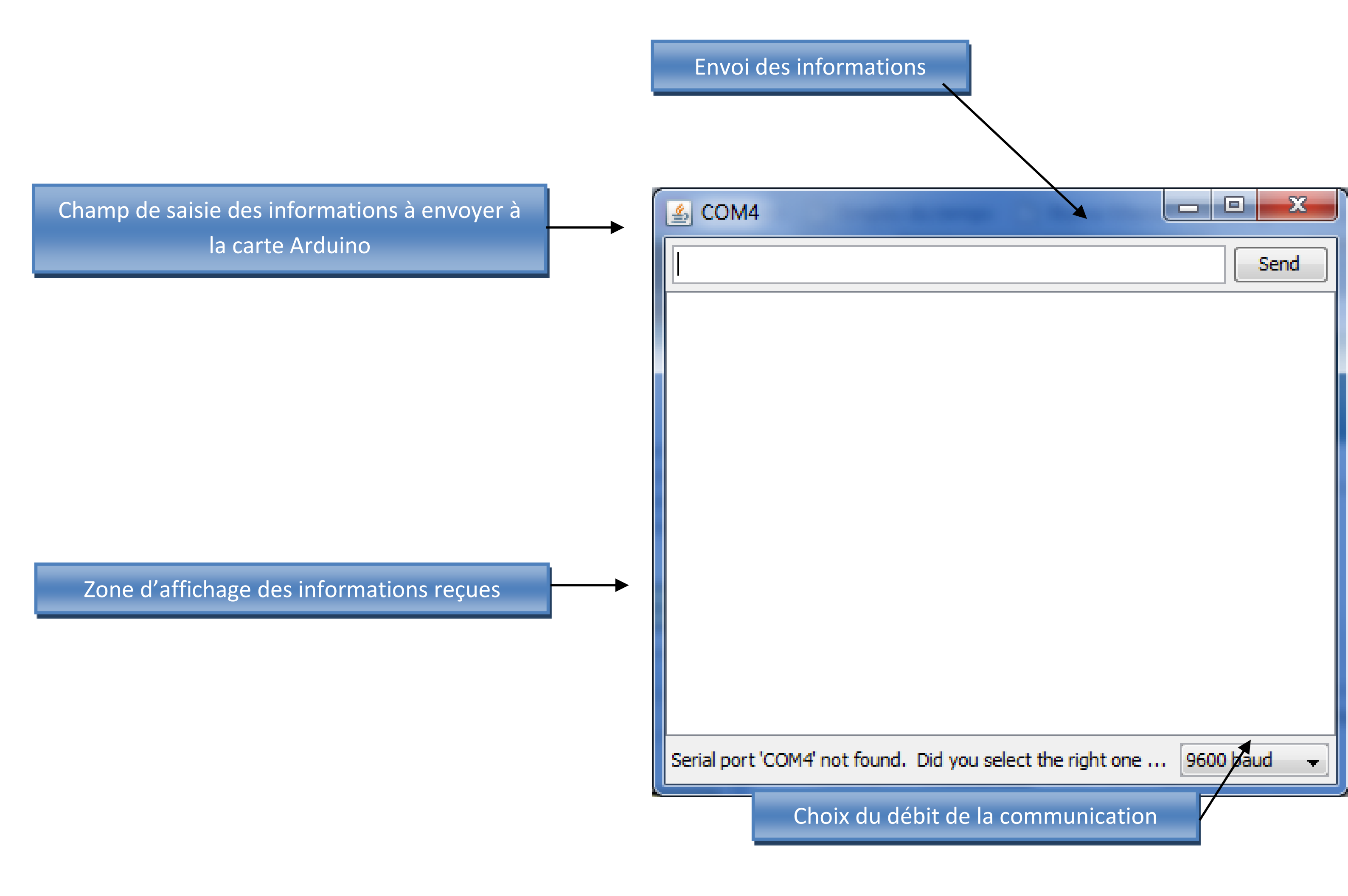
Nous avions déjà quelques bases en langage C grâce à nos cours de programmation au 1er semestre d'Ei2, ce qui nous a permis de nous familiariser assez rapidement avec le logiciel et donc de développer le programme de notre robot.

Espace de développement intégré



Terminal série

Un terminal série est intégré dans le logiciel Arduino. Il permet l'affichage de messages textes en provenance la carte Arduino et l'envoi de caractères vers celle-ci. Le terminal permet de facilité la mise au point des programmes, en affichant sur le PC l'état des variables, de résultats de conversions Analogique-Numérique. Il est essentiel pour améliorer, tester (« debugger ») et améliorer ses programmes.



Transférer des programmes vers la carte Arduino

1. *Saisie du programme et vérification du code*

On écrit le programme et on vérifie si la syntaxe est correcte en le compilant (bouton *Verify*). Si tout est correct, aucun message d'erreur n'apparaît dans la console et « *Done compiling* » s'affiche, indiquant que la vérification s'est déroulée sans problème.



2. *Sélection du port et de la carte*

Avant de transférer le programme compilé, il faut tout d'abord sélectionner la carte Arduino que l'on utilise (menu *Tools* → *Boards*). On pourra choisir COM 1 ou 2 pour une carte série, et COM 4, 5, 7 et plus pour une carte USB.

3. *Transfert du programme*

Une fois le port et la carte sélectionnés, il suffit de cliquer sur le bouton U*pload :*
La carte va se réinitialiser automatiquement et le transfert va démarrer. Les LEDs des lignes TX et RX de la carte Arduino vont clignoter, indiquant que le programme est bien transféré. Le message » *Done uploading »* confirmera le bon transfert du programme.

4. *Programme transféré*

Lorsque l'on transfert un programme, en utilisant le bootloader Arduino, un code binaire a été chargé dans le µ-contrôleur sur la carte. Ceci permet de transférer le programme sans avoir besoin d'aucun matériel externe. Quand le transfert est terminé, le bootloader est actif pendant un petit temps (il écoute pour voir si un nouveau programme arrive) une fois que la carte est réinitialisée à la fin du transfert du programme ; et le dernier programme stocké dans la carte s'exécute.

2. Explication du code

Déclarations

```
void setup() {

Serial1.begin(9600);
int vitesse(int, int);
int transfo(int *chaine, int *tableauvit);

}
```

Nous initialisons le port série 1 Le port série est initialisé à 9600 bits/seconde. Pour ceci, nous utilisons l'instruction begin. Nous déclarons les deux fonctions, vitesse et transfo, qui vont nous servir à traiter les données reçues et les envoyées sur les moteurs.

Réception des données

```
while (Serial1.available()>0) {

for (j=0; j<6; j++){

 delay(10);
 Serial.print(kap[j]-48);

}
```

La carte reçoit 6 caractères, ces sont des chiffres codés en Ascii. Une fois que la carte reçoit quelque chose sur son port série 1, la carte lit les caractères un par un avec un intervalle de temps de 10 ms. Nous utilisons l'instruction read pour lire les caractères qui sont sur le port série 1. Cet intervalle est plus petit que l'intervalle de temps d'envoi des données. Les caractères sont gardés en mémoire dans un tableau.

Une fois que ces six chiffres sont traités, la carte enregistre les suivants et ainsi de suite.

Traitement des données

Les données reçues par la carte seront traitées avant d'être envoyées aux moteurs. En effet, les trois premiers caractères représentent la vitesse de la roue droite, les trois suivants représentent la vitesse de la roue gauche. Pour décomposer, traiter et envoyer sur les moteurs les données, nous avons programmé deux fonctions ; la fonction 'vitesse' et la fonction ' transfo'.

La fonction 'transfo'

```
int transfo(int *chaine, int *tableauvit){
tableauvit[0] = (chaine[0]-48)*100+(chaine[1]-48)*10+(chaine[0]-48) - 355;
tableauvit[1] = (chaine[3]-48)*100+(chaine[4]-48)*10+(chaine[5]-48) - 355;
}
```

Cette fonction reçoit en entrée deux chaines de caractères.

Comme nous l'avons mentionné avant, les trois premières cases représentent la vitesse de la roue droite. En effet, le premier chiffre représente les centaines, le second représente les dizaines et le troisième représente les unités. La carte reçoit les chiffres en Ascii, il faut, donc les décaler. Nous savons que le code Ascii de 1 est 48. Nous décalons, alors, les chiffres de 48. D'autre part, la valeur de chaque vitesse est décalée de +355 (voir partie Programmation express 2008 – explication du code), il faut donc la décaler de -355. Une fois tous ces chiffres décalés, multipliés par leur coefficient (100 ou 10 ou 1), ils sont sommés et gardé en mémoire dans la première case du tableau *'tableauvit'*. Les mêmes opérations sont répétées pour les valeurs suivantes, pour obtenir la vitesse de la roue gauche.

La fonction vitesse

int vitesse (int vitD, int vitG)

Cette fonction reçoit en entrée deux variables ; vitesse droite, vitesse gauche. Les valeurs reçues par la fonction sont comprises dans l'intervalle [-255 ; 255]. La vitesse du moteur est codée sur un octet, donc ; V moteur = [0 ;255] * coeff.

On distingue 5 cas différents pour le contrôle du robot :

- si la vitesse droite & la vitesse gauche sont nulles :

  ➔ on ne transmet rien sur les moteurs.

- sinon, si la vitesse droite & la vitesse gauche sont supérieures à 0 :

  ➔ on transmet à chaque moteur la vitesse qui lui correspond, et on règle la direction des moteurs vers l'avant (*High*).

- sinon, si la vitesse gauche est inférieure à 0 :

  ➔ on règle la vitesse du moteur gauche vers l'arrière (*Low*).

- si la vitesse droite est < 0 et la vitesse gauche > 0 :

  ➔ on fait l'inverse.

- et si les deux vitesses sont < 0 :

  ➔ on règle la direction des 2 moteurs vers l'arrière.

d. Conclusion

Ce projet nous a permis de développer nos compétences en Visual Basic et en C. En effet, le groupe composé Khalil et Omar a pu découvrir la communication via le port Com et les différents outils (Windows Form) proposés par Visual Basic. Le binôme composé par Pierre et Abdelkarim a pu découvrir le protocole pour établir une communication via le port COM en C et les différentes fonctions propres à une carte Arduino. En fin, les deux binômes ont pu consolider les structures logiques soit en C soit en Visual Basic.

D'autre part, nous avons pu approfondir nos connaissances en électronique et en électrotechnique. Initialement, le robot était alimenté avec des piles mais cela ne suffisait ; le robot ne roule pas, il faut un courant important au démarrage que les piles ne débitent pas. Nous avons, donc, choisi une batterie adéquate au robot.

Au cours de ce projet, nous avons rencontré quelques problèmes soit d'ordre technique soit d'ordre organisationnel. En effet, nous avons rencontré des problèmes avec une carte Arduino ; pendant 3 séances nous avons effectué des tests sur le port série d'une carte Duemilanove qui n'ont pas aboutis, nous avons été dans l'obligation de changer de carte. Nous avons utilisé, par la suite, une carte Arduino Mega. D'autre part, une des deux cartes FM est tombée en panne durant le projet, nous avons donc commandé une autre carte que nous avons reçue tardivement.

Dans l'ensemble, ce projet était bénéfique pour le groupe.

Nous avons eu l'idée avec Mr Lagrange d'exposer Wii'Stia à la journée porte ouverte; ça attirera les curieux et surtout ça montrera l'aboutissement d'un projet réalisé complètement par des étudiants.

Pour finir, nous nous sommes rendus compte au cours de ce projet qu'il y avait d'autres pistes à explorer dans la même voie ;

- Remplacer le robot par une voiture, un avion
- Commander un robot avec une WiiMote

Pour finir, nous proposons de monter un club de robotique au sein de l'Istia. En effet, un tel club pourra initier les étudiants aux nouvelles technologies, développer leurs sens de créativité et apprendre en dehors du cadre d'un cours.

e. Annexes

Code sur VB Express

```
/////////////////////////////////////////////////////////////////////////////
/////
'     MultipleWiimoteForm.cs
'     Managed Wiimote Library Tester
'     Written by Brian Peek (http://www.brianpeek.com/)
'  for MSDN's Coding4Fun (http://msdn.microsoft.com/coding4fun/)
'     Visit
http://blogs.msdn.com/coding4fun/archive/2007/03/14/1879033.aspx
'  and http://www.codeplex.com/WiimoteLib
'  for more information
'////////////////////////////////////////////////////////////////////////////
//////


Imports Microsoft.VisualBasic
Imports System
Imports System.Drawing
Imports System.Drawing.Imaging
Imports System.Windows.Forms
Imports WiimoteLib

Namespace WiimoteTest


    'Déclaration de variables

    Partial Public Class WiimoteInfo
        Inherits UserControl
        Private Delegate Sub UpdateWiimoteStateDelegate(ByVal args As
WiimoteChangedEventArgs)
        Private Delegate Sub UpdateExtensionChangedDelegate(ByVal args As
WiimoteExtensionChangedEventArgs)


        Dim com As System.IO.Ports.SerialPort
        Private b As New Bitmap(256, 192, PixelFormat.Format24bppRgb)
        Private g As Graphics
        Private mWiimote As Wiimote
        Dim a, e, c, d, f As Long
        Dim pourg As String
        Dim pourd As String


        Public Sub New() 'charger l'image de la wiifit
            InitializeComponent()
            g = Graphics.FromImage(b)
        End Sub


        Public Sub New(ByVal wm As Wiimote) 'initialisation de la wiifit
ici on va l'appelé wm
            Me.New()
```

```
            mWiimote = wm
        End Sub

        Public Sub UpdateState(ByVal args As WiimoteChangedEventArgs) 'code
récuperé sur coding4fun qui utilise la librarie wii
            BeginInvoke(New UpdateWiimoteStateDelegate(AddressOf
UpdateWiimoteChanged), args)
        End Sub

        Public Sub UpdateExtension(ByVal args As
WiimoteExtensionChangedEventArgs) 'code récuperé sur coding4fun qui utilise
la librarie wii
            BeginInvoke(New UpdateExtensionChangedDelegate(AddressOf
UpdateExtensionChanged), args)
        End Sub
        ' changement a effectuer quand l'utilisateur change de niveau de
sensibilité du robot
        Private Sub RadioButton1_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
RadioButton1.CheckedChanged
            f = 100 'facile
        End Sub

        Private Sub RadioButton2_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
RadioButton2.CheckedChanged
            f = 170 'intermediaire
        End Sub

        Private Sub RadioButton3_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
RadioButton3.CheckedChanged
            f = 255 ' expert
        End Sub
        Private Sub RadioButton4_CheckedChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
RadioButton4.CheckedChanged
            f = 70 'initiation
        End Sub
        Private Sub TextBox1_TextChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TextBox1.TextChanged
            If TextBox1.Text = "" Then  ' la condition c'est  pour eviter
l'interuption de l'envoie des données au robot quand on efface la case
TextBox1
                f = 0
            Else
                f = TextBox1.Text
            End If
        End Sub

        Private Sub UpdateWiimoteChanged(ByVal args As
WiimoteChangedEventArgs)
            Dim ws As WiimoteState = args.WiimoteState



            Select Case ws.ExtensionType ' choix de la balance board comme
outil et initialisé les variable
                ' Dans la cas où la wiifit sera branché on affiche les
pressions sur les capteurs et le poid
                'en kg ou en pound celon le choix de l'utilisateur
```

```
                Case ExtensionType.BalanceBoard
                    If chkLbs.Checked Then
                        lblBBTL.Text =
ws.BalanceBoardState.SensorValuesLb.TopLeft.ToString()
                        lblBBTR.Text =
ws.BalanceBoardState.SensorValuesLb.TopRight.ToString()
                        lblBBBL.Text =
ws.BalanceBoardState.SensorValuesLb.BottomLeft.ToString()
                        lblBBBR.Text =
ws.BalanceBoardState.SensorValuesLb.BottomRight.ToString()
                        lblBBTotal.Text =
ws.BalanceBoardState.WeightLb.ToString()
                        Poid.Text =
ws.BalanceBoardState.WeightLb.ToString()
                    Else
                        lblBBTL.Text =
(ws.BalanceBoardState.SensorValuesKg.TopLeft.ToString())
                        lblBBTR.Text =
(ws.BalanceBoardState.SensorValuesKg.TopRight.ToString())
                        lblBBBL.Text =
ws.BalanceBoardState.SensorValuesKg.BottomLeft.ToString()
                        lblBBBR.Text =
ws.BalanceBoardState.SensorValuesKg.BottomRight.ToString()
                        lblBBTotal.Text =
ws.BalanceBoardState.WeightKg.ToString()
                        Poid.Text =
ws.BalanceBoardState.WeightKg.ToString()
                    End If
                    lblCOG.Text =
ws.BalanceBoardState.CenterOfGravity.ToString() 'position du centre de
gravité de la wiifit
            End Select
            ' pour raison des operations qu'on va effectuer sur les
pressios des capteurs après
            ' on les affecte à des differentes variables
            a = ws.BalanceBoardState.SensorValuesKg.TopLeft
            e = ws.BalanceBoardState.SensorValuesKg.TopRight
            c = ws.BalanceBoardState.SensorValuesKg.BottomLeft
            d = ws.BalanceBoardState.SensorValuesKg.BottomRight

            'on mets un seuil aux pressions pour ne pas depacer 255 puisque
en RGB on peut pas depacer ce seuil.
            If a > 255 Then
                a = 255
            End If
            If e > 255 Then
                e = 255
            End If
            If c > 255 Then
                c = 255
            End If
            If d > 255 Then
                d = 255
            End If

            g.Clear(Color.Black)
            'elelver les petites virgules qui sont proches du 0 '

            If ws.BalanceBoardState.SensorValuesKg.TopLeft < 0 Then
                ws.BalanceBoardState.SensorValuesKg.TopLeft = 0
```

```
            End If
            If ws.BalanceBoardState.SensorValuesKg.TopRight < 0 Then
                ws.BalanceBoardState.SensorValuesKg.TopRight = 0
            End If
            If ws.BalanceBoardState.SensorValuesKg.BottomLeft < 0 Then
                ws.BalanceBoardState.SensorValuesKg.BottomLeft = 0
                If ws.BalanceBoardState.SensorValuesKg.BottomRight < 0 Then
                    ws.BalanceBoardState.SensorValuesKg.BottomRight = 0
                End If
            End If


            'definir les pourecentages qu'ils faut envoyer au robot.(voir
compte rendu)et les afficher
            pourg =
Int(((Math.Abs(ws.BalanceBoardState.SensorValuesKg.TopLeft) -
Math.Abs(ws.BalanceBoardState.SensorValuesKg.BottomLeft))) /
(Math.Abs(ws.BalanceBoardState.SensorValuesKg.TopLeft) +
Math.Abs(ws.BalanceBoardState.SensorValuesKg.BottomLeft) +
Math.Abs(ws.BalanceBoardState.SensorValuesKg.TopRight +
Math.Abs(ws.BalanceBoardState.SensorValuesKg.BottomRight))) * f) + 355
            pourd =
Int(((Math.Abs(ws.BalanceBoardState.SensorValuesKg.TopRight) -
Math.Abs(ws.BalanceBoardState.SensorValuesKg.BottomRight))) /
(Math.Abs(ws.BalanceBoardState.SensorValuesKg.TopLeft) +
Math.Abs(ws.BalanceBoardState.SensorValuesKg.BottomLeft) +
Math.Abs(ws.BalanceBoardState.SensorValuesKg.TopRight +
Math.Abs(ws.BalanceBoardState.SensorValuesKg.BottomRight))) * f) + 355
            ' pourg =
Int(((Math.Abs(ws.BalanceBoardState.SensorValuesKg.TopLeft) -
Math.Abs(ws.BalanceBoardState.SensorValuesKg.BottomLeft))) /
(Math.Abs(ws.BalanceBoardState.SensorValuesKg.TopLeft) +
Math.Abs(ws.BalanceBoardState.SensorValuesKg.BottomLeft) +
Math.Abs(ws.BalanceBoardState.SensorValuesKg.TopRight +
Math.Abs(ws.BalanceBoardState.SensorValuesKg.BottomRight))) * 255) + 355
            'pourd =
Int(((Math.Abs(ws.BalanceBoardState.SensorValuesKg.TopRight) -
Math.Abs(ws.BalanceBoardState.SensorValuesKg.BottomRight))) /
(Math.Abs(ws.BalanceBoardState.SensorValuesKg.TopLeft) +
Math.Abs(ws.BalanceBoardState.SensorValuesKg.BottomLeft) +
Math.Abs(ws.BalanceBoardState.SensorValuesKg.TopRight +
Math.Abs(ws.BalanceBoardState.SensorValuesKg.BottomRight))) * 255) + 355
            textg.Text = pourg
            textd.Text = pourd
            'vu que la balance board detecte une petite pression meme si il
n'y a perssone donc
            'on mets les pourcentages à 0(355 dans notre échèle voir compte
rendu) si le poid est en dessous de 10
            If ws.BalanceBoardState.WeightKg < 10 Then
                pourg = 355
                pourd = 355
            End If





            ''cette partie correspond a l'affichage de la wii et
modelisation avec couleur des differentes pressions
            If a > 10 Then
```

```
            Panel1.BackColor = Color.FromArgb(0, 0, 255 - a)
        Else
            Panel1.BackColor = Color.White


        End If
        If c > 10 Then
            Panel3.BackColor = Color.FromArgb(0, 0, 255 - c)


        Else
            Panel3.BackColor = Color.White


        End If
        If e > 10 Then
            Panel2.BackColor = Color.FromArgb(0, 0, 255 - e)


        Else
            Panel2.BackColor = Color.White

        End If
        If d > 10 Then
            Panel4.BackColor = Color.FromArgb(0, 0, 255 - d)


        Else
            Panel4.BackColor = Color.White


        End If
        'code récuperé sur coding4fun qui utilise la librarie wii pour
donner des infos sur l'etat de la batterieet la wiifit
        If ws.Battery > &HC8 Then
            pbBattery.Value = (&HC8)
        Else
            pbBattery.Value = (CInt(Fix(ws.Battery)))
        End If

        lblBattery.Text = ws.Battery.ToString()
        lblDevicePath.Text = "Device Path: " & mWiimote.HIDDevicePath

    End Sub


    'code récuperé sur coding4fun qui utilise la librarie wii
    Private Sub UpdateExtensionChanged(ByVal args As
WiimoteExtensionChangedEventArgs)
        chkExtension.Text = args.ExtensionType.ToString()
        chkExtension.Checked = args.Inserted
    End Sub
    'code récuperé sur coding4fun qui utilise la librarie wii
    Public WriteOnly Property Wiimote() As Wiimote
        Set(ByVal value As Wiimote)
            mWiimote = value
        End Set
    End Property
    ' cette patie du code correspond à la combobox du choix de la
vitesse
```

```
        Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ComboBox1.SelectedIndexChanged
            If ComboBox1.SelectedIndex = 0 Then
                RS232.BaudRate = 600
            End If
            If ComboBox1.SelectedIndex = 1 Then
                RS232.BaudRate = 1200
            End If
            If ComboBox1.SelectedIndex = 2 Then
                RS232.BaudRate = 2400
            End If
            If ComboBox1.SelectedIndex = 3 Then
                RS232.BaudRate = 4800
            End If
            If ComboBox1.SelectedIndex = 4 Then
                RS232.BaudRate = 9600
            End If
            If ComboBox1.SelectedIndex = 5 Then
                RS232.BaudRate = 14400
            End If
            If ComboBox1.SelectedIndex = 6 Then
                RS232.BaudRate = 19200
            End If
            If ComboBox1.SelectedIndex = 7 Then
                RS232.BaudRate = 28800
            End If
            If ComboBox1.SelectedIndex = 8 Then
                RS232.BaudRate = 38200
            End If
            If ComboBox1.SelectedIndex = 9 Then
                RS232.BaudRate = 56000
            End If
            If ComboBox1.SelectedIndex = 10 Then
                RS232.BaudRate = 57600
            End If
            If ComboBox1.SelectedIndex = 0 Then
                RS232.BaudRate = 115200
            End If
            If ComboBox1.SelectedIndex = 0 Then
                RS232.BaudRate = 128000
            End If
            If ComboBox1.SelectedIndex = 0 Then
                RS232.BaudRate = 258000
            End If
            If ComboBox1.SelectedIndex = 4 Then
                RS232.BaudRate = 9600
            End If
        End Sub

        Private Sub WiimoteTest_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
            Dim ports() As String = IO.Ports.SerialPort.GetPortNames() '
tableau de string acceuillant les ports present sur la machine

            'configuration du port RS232
            'RS232.BaudRate defini précedement
            RS232.Parity = IO.Ports.Parity.None 'pas de parité
            RS232.StopBits = IO.Ports.StopBits.One 'un bit d'arrêt par
octet
            RS232.DataBits = 8 'nombre de bit par octet
```

```
            RS232.RtsEnable = True 'ligne Rts activé
            RS232.DtrEnable = True 'ligne Dtr activé

            Choix_Port.Items.AddRange(ports) ' on ajoute le nom des ports
dans le combobox

            Try

                Choix_Port.SelectedIndex = 0 ' on donne le focus au premier
port du combobox

            Catch ' message d'erreur si aucun port COM n'est détecté sur la
machine et on ferme l'application

                MsgBox("Il semble ne pas y avoir de port RS232 sur votre
machine, cette application ne pourra pas marcher")
                Application.Exit()

            End Try
        End Sub
        'cette partie du code correspond à la programmation du port serie
et la connexion robot
        Private Sub Connexion_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Connexion.Click

            Button1.Enabled = False
            Button2.Enabled = False

            If Connexion.Text = "Connexion" Then

                Try

                    'on ouvre le port désiré
                    RS232.PortName = Choix_Port.Text
                    RS232.Open()

                Catch ' message d'erreur si l'on ne peut pas ouvrir le port

                    MsgBox("Il semble que le port choisi est déjà en cours
d'utilisation par une autre application, merci de choisir un port libre
pour se connecter")
                    Exit Sub

                End Try

                Vitesse.Text = RS232.BaudRate & Space(1) & "Bauds" 'on
affiche la vitesse du port
                Connexion.Text = "Déconnexion" ' on change l'intitulé du
bouton
                Choix_Port.Enabled = False ' on désactive le combobox
pendant l'utilisation du port
            Else

                Vitesse.Text = "" 'on efface la vitesse
                Connexion.Text = "Connexion" ' on change l'intitulé du
bouton
                RS232.Close() 'on ferme le port
                Choix_Port.Enabled = True 'on réactive le combobox

            End If
            If Connexion.Text = "Déconnexion" Then
```

```
                Button1.Enabled = True
                Button2.Enabled = True

            End If
        End Sub
        'ici on initialise un timer qui va envoyé les 2 pourcentages aux
robot chaques 100 ms
        Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer1.Tick

            Try
                RS232.Write(pourg & pourd) 'ecrire dans le port serie
                'convertion des 2 pourcentage en chaine de char
                pourg = String.Empty
                pourd = String.Empty
            Catch ex As Exception
                MsgBox(ex.ToString)
            End Try
        End Sub
        ' boutton pour se connecter du robot en activant le timer
        Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

            Connexion.Enabled = False
            Timer1.Enabled = True

        End Sub
        ' boutton pour se déconnecter du robot en désactivant le timer...
        Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
            Timer1.Enabled = False
            RS232.Write(355 & 355) ' arretez le robot
            Connexion.Enabled = True
        End Sub


    End Class

End Namespace
```

Code en C sur Arduino

```
#define PwmPinMotorD 10

#define PwmPinMotorG 11

#define DirectionPinMotorD 12

#define DirectionPinMotorG 13
```

```
 void setup() {


 Serial1.begin(9600);

 int vitesse(int, int);

 int transfo(int *chaine, int *tableauvit);

   }

 void loop(){

int kap[100];

int vat[2];

 int i ;

 int j;

int vitD; int vitG;

 while (Serial1.available()>0) {

 for (j=0; j<2; j++){

   delay(10);

                    }

transfo(&kap[0],&vat[0]);

vitesse(vat[0],vat[1]);

 }

  }


int vitesse(int vitD, int vitG){

 pinMode(PwmPinMotorD, OUTPUT);

 pinMode(PwmPinMotorG, OUTPUT);

 pinMode(DirectionPinMotorD, OUTPUT);

 pinMode(DirectionPinMotorG, OUTPUT);
```

```
if (vitD == 0 && vitG == 0){
        analogWrite(PwmPinMotorD, 0);
        digitalWrite(DirectionPinMotorD, HIGH);
        analogWrite(PwmPinMotorG, 0);
        digitalWrite(DirectionPinMotorG, HIGH);
           }

else          {
         if (vitD >= 0) {

            if (vitG >= 0){
                   analogWrite(PwmPinMotorD, vitD);
                   digitalWrite(DirectionPinMotorD, HIGH);
                   analogWrite(PwmPinMotorG, vitG);
                   digitalWrite(DirectionPinMotorG, HIGH);
                   }

            else       {
                    analogWrite(PwmPinMotorD, vitD);
                    digitalWrite(DirectionPinMotorD, HIGH);
                    analogWrite(PwmPinMotorG, -vitG);
                    digitalWrite(DirectionPinMotorG, LOW);
                    }

            }
```

```
else {
        if (vitG >= 0){
        analogWrite(PwmPinMotorD, -vitD);
        digitalWrite(DirectionPinMotorD, LOW);
        analogWrite(PwmPinMotorG, vitG);
        digitalWrite(DirectionPinMotorG, HIGH);
        }

    else {
        analogWrite(PwmPinMotorD, -vitD);
        digitalWrite(DirectionPinMotorD, LOW);
        analogWrite(PwmPinMotorG, -vitG);
        digitalWrite(DirectionPinMotorG, LOW);
      }
 }
    }
}

int transfo(int *chaine, int *tableauvit){
tableauvit[0] = (chaine[0]-53)* 50;
tableauvit[1] = (chaine[1]-53)*50 ;
}
```